構文的翻訳図式（SDTS）を用いたプログラムからのプログラム言語の翻訳規則の学習

大内 章[†] 今田 圭太[‡]
東京電機大学院 理工学研究科[‡‡]

中村 克彦[†‡]
東京電機大学 理工学部[†‡‡]

1 まえがき

われわれは2000年から文脈自由文法（CFG : context free grammar）の文法推論についての研究を続けており，独自の CFG の漸次学習システム Synapse の開発と改良を重ねてきた．Synapse は，ある言語に含まれる記号列 (正例) の集合と，含まれない記号列 (負例) の集合から，これらを満足する規則集合 (文法) を合成する [1]．現在，Synapse は確定節文法（DCG：Definite Clause Grammar）の学習と，これを応用した構文的翻訳図式（SDTS：syntax directed translation schema）[2] の学習が可能なように拡張されている．

この研究の目的は，SDTS の学習を以下のような帰納推論に応用することである．

- プログラムの例からの文法の生成
- プログラム変換例からのトランスレータの学習．および，この特別な場合であるプログラム言語のコンパイラ（中間言語への変換）の学習．

この報告では，CFG と DCG および SDTS の関係について述べ，コンパイラの学習の例として実際に C 言語の算術式と if 文の FORTH 風中間言語への変換を示す．

2 CFG と DCG および SDTS

CFG は $p \to u$ なる形式の (生成) 規則の集合によって定義される．ここで，p は非終端記号であり，u は終端記号と非終端記号とからなる記号列である．CFG には開始記号 s という特別な非終端記号があり，s から規則の適用によって記号列が導出される．

DCG は論理プログラミングにもとづいており，CFG の生成規則に $p(T)$ の形で特別な項（DCG 項）T を付加した非終端記号を含むように拡張した文法である．DCG 項 T によって規則の適用の制限や部分的に導出される記号列の操作などを表わせる．このため DCG は，CFG よりも複雑な言語を表わすことができる．DCG の規則は機械的に Horn 節に変換でき，構文解析および文字列の生成を行う Prolog プログラムとして直接実行することが可能である．

表 1: DCG と SDTS の対応 (a : 入力記号の終端記号 b : 出力記号の終端記号 p,q,r : 共通の非終端記号)

SDTS	DCG
$p \to a, b$	$p([b\|X]/X) \to a$
$p \to ar, r$	$p(X/Y) \to a, r(X/Y)$
$p \to ra, r$	$p(X/Y) \to r(X/Y), a$
$p \to r, br$	$p(X/Y) \to r([b\|X]/Y)$
$p \to r, rb$	$p(X/Y) \to r(X/[b\|Y])$
$p \to ar, rb$	$p(X/Y) \to a, r(X/[b\|Y])$
$p \to ar, br$	$p(X/Y) \to a, r([b\|X]/Y)$
$p \to ra, rb$	$p(X/Y) \to r(X/[b\|Y]), a$
$p \to ar, br$	$p(X/Y) \to a, r(X/[b\|Y])$
$p \to qr, qr$	$p(X/Z) \to q(X/Y), r(Y/Z)$
$p \to qr, rq$	$p(X/Z) \to q(Y/Z), r(X/Y)$

SDTS は，入力言語と出力言語の二つの文脈自由文法の構文規則を対応付けた $A \to u, v$ の形式の規則の集合によって定義される．ここで，p は非終端記号であり，u と v は終端記号と非終端記号とからなる記号列である．ただし，u に含まれる非終端記号の集合は v に含まれる非終端記号の集合と一致しなければならない．SDTS の規則導出によって，開始記号 s から入力と出力に対応する二つの記号列が同時に導出される．

DCG によって SDTS の規則をあらわすことができる．表 1 に，DCG の SDTS への対応例を示す．この DCG 規則のすべての非終端記号は X/Y の形で表した差分リストを DCG 項として必ずもっている．

このような DCG の規則によって，SDTS の開始記号 s から $a_1\ a_2\ \ldots\ a_m,\ b_1\ b_2\ \ldots\ b_n$ が導出されるとき，

Learning Translation Rules of Programmig Languages from Programs using SDTS
Akira Oouchi[†], Keita Imada[‡], Katsuhiko Nakamura[†‡]
Department of Science and Engineering, Graduate School of Tokyo Denki University[‡‡], Department of Science and Engineering, Tokyo Denki University[†‡‡]
, 350-0394 Ishizaka Hatoyama-machi Hiki-gun Saitama-ken Japan[‡‡ †‡‡]
07smk04@ms.dendai.ac.jp[†] imada@naklab.k.dendai.ac.jp[‡]
nakamura@k.dendai.ac.jp[†‡]

合成した文法
$n \rightarrow n\ q,\ q\ n$
$p \rightarrow op1\ n\ ,\ n\ op1$
$q \rightarrow op2\ n\ ,\ n\ op2$
$s \rightarrow n\ ,\ n \mid s\ p\ ,\ s\ p$
初期規則
$op1 \rightarrow +\ ,\ + \mid -\ ,\ -$
$op2 \rightarrow *\ ,\ * \mid /\ ,\ /$
$n \rightarrow a\ ,\ a \mid b\ ,\ b$

図 1: 算術式から逆ポーランド記法の式への変換規則

DCG の開始記号 s($[b_1\ b_2\ \ldots\ b_n]/[]$) から $a_1\ a_2\ \ldots\ a_m$ への変換が可能である．ただし，$m > n$ または $m = n$ または $m < n$ である．

3 中間言語

FORTH は，演算と命令がすべて逆ポーランド記法とその拡張で表されるプログラム言語である [3]．この特徴はバイトコードや P コードなどの中間言語と共通している．さらに，FORTH は制御命令にラベルを使用せず，FORTH の条件分岐文は " *cond* IF *exe* THEN " のように表される．*cond* とは条件式を，*exe* は条件が真であったときに実行される文をそれぞれ示している．これらの理由より，FORTH は SDTS の出力言語に適している．

4 SDTS を用いた算術式の翻訳

コンパイラ学習の第一の例は，かっこを含まない算術式から逆ポーランド記法の式への変換である．Synapse で SDTS 規則を学習する場合，対応する入力記号列の集合と出力記号列の対の集合を正例に，対応しない記号列の集合を負例として与える必要がある．次に，与えた正例と負例の一部を次に示す．

正例 : s([a,+,a]/[]) $\stackrel{*}{\Rightarrow}$ [a,a,+]
s([a,*,b,+,a]/[]) $\stackrel{*}{\Rightarrow}$ [a,b,*,a,+]
負例 : s([a,+,a]/[]) $\stackrel{*}{\Rightarrow}$ [a,+,a]
s([a,+,a,+,a]/[]) $\stackrel{*}{\Rightarrow}$ [a,a,a,+,+]

実際に Synapse に与えた正例は約 400 個，負例は 20 個である．図 1 に学習した文法を示す．図 1 中の初期規則はあらかじめ与えられた規則をあらわしている．

これらの変換規則の学習に要した時間は，Athron64 X2 2.20GHz の CPU を用いて約 14 秒であった．

$w \rightarrow if, \mathtt{IF}$	$q \rightarrow t\ u, t\ u$
$p \rightarrow w\ r, r\ w$	$s \rightarrow p\ q, p\ q$
$t \rightarrow '\{'\ exe, exe$	$r \rightarrow '('\ v, v$
$u \rightarrow '\}'\ then, \mathtt{THEN}$	$v \rightarrow cond\ ')', cond$

図 2: if 文から中間言語の条件分岐文の変換規則

5 SDTS を用いた if 文の中間言語への翻訳

SDTS を用いて if 文を中間言語に変換するために与えた正例と負例の一部を次に示す．

正例 :
s(['if','(','cond',')','','exe',''']/[])
$\stackrel{*}{\Rightarrow}$ ['cond','if','exe','then']

負例 :
s(_) $\stackrel{*}{\Rightarrow}$ ['if','cond','exe']
s(_) $\stackrel{*}{\Rightarrow}$ ['exe','exe','cond','exe','exe']

Synapse に与えた正例は上の一つのみであるが，負例は約 40 個である．図 2 に学習した C 言語の if 文から FORTH 風中間言語の変換規則を示す．時間は Athron64 X2 2.20GHz で図 2 内の二つの規則を 0.22 秒で合成した．規則数が増えると変換規則学習に多くの例と時間を要するためこれ以上の数の規則の合成はできていない．

6 むすび

この報告では，Synapse と SDTS の関係を述べ，実際に SDTS を用いた変換規則の学習を結果を示した．

今後の課題として，プログラムの変数宣言部などを中間言語へ変換するための規則の学習，より広範囲なコンパイラの学習の二つがあげられる．変数の宣言に関する制限は文脈自由で表せない．これをどのように解決するかがコンパイラ学習において重要な問題である．

参考文献

[1] Katsuhiko Nakamura : Incremental Learning of Context Free Grammars by Bridging Rule Generation and Search for Semi-optimum Rule Sets , pp. 72-73 in Lecture Notes in Computer Science, ICGI, 2006.

[2] Aho, A. V. and Ullman, J. D. : 9. 2 Syntax-directed translations in The Theory of Parsing, Translation, and Compiling , ACM, 1972.

[3] Forth Interest Group URL : http://www.forth.org/ .